Аношенко Даниил

ВАРИАНТ 1

# Дана схема базы данных в виде следующих отношений. С помощью операторов SQL создать логическую структуру соответствующих таблиц для хранения в СУБД, используя известные средства поддержания целостности (NOT NULL, UNIQUE, и т.д.). Обосновать выбор типов данных и используемые средства поддержания целостности. При выборе подходящих типов данных использовать информацию о конкретных значениях полей БД.

CREATE TABLE customer (

id int primary key check(id>0), surname text not null,

district varchar(30),

discount int check(discount >= 0)

);

* + id во всех таблицах является первичным ключом, то есть должен быть уникальным и не отрицательным
  + name/surname является основной информацией об объекте поэтому поле всегда должно быть заполнено
  + district, едва ли название района превысит 30 символов
  + discount не может быть отрицательным, но может быть равен 0

CREATE TABLE shop (

id int primary key check(id>0), name text not null,

district varchar(30), fee int check(fee >= 0)

);

* + fee не может быть отрицательным, но может быть равен 0

CREATE TABLE book (

id int primary key check(id>0), name text not null,

cost int check(cost >= 0), district varchar(30),

amount int check(amount >= 0)

);

* + amount не может быть отрицательным, но может быть равен 0

CREATE TABLE purchase (

number int primary key check(number>0), month varchar(9),

id\_shop int references shop(id), id\_customer int references customer(id), id\_book int references book(id),

amount int check(amount >= 0), total int check(total>=0)

);

* + так как дата в исходных таблицах представлена только месяцем, её не удастся хранить в типе данных date. Самое длинное название месяца на английском = 9 символам
  + id\_shop/id\_customer/id\_book - внешние ключи, которые ссылаются на соответствующие id в своих таблицах
  + amount не может быть отрицательным, но может быть равен 0
  + total не может быть отрицательным, но может быть равен 0

# Ввести в ранее созданные таблицы конкретные данные (см. прил. 1). Использовать скрипт-файл из операторов INSERT или вспомогательную утилиту.

INSERT INTO customer VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (001, | 'Cидоров', | 'Нижегородский', | 10), |
| (002, | 'Потапов', | 'Советский', 20), |  |
| (003, | 'Попов', | 'Ленинский',10), |  |
| (004, | 'Романова', | 'Нижегородский', | 10), |
| (005, | 'Миронов', | 'Автозаводский', | 15), |
| (006, | 'Попов', | 'Советский', 0); |  |
| INSERT INTO shop VALUES | | | |
| (001, | 'Знание', 'Автозаводский', 7), | | |
| (002, | 'Наука', 'Нижегородский', 8), | | |
| (003, | 'Книжный мир', 'Приокский',6), | | |
| (004, | 'Книги', 'Сормовский', 9), | | |
| (005, | 'Книги', 'Советский', 7); | | |

INSERT INTO book VALUES

|  |  |  |  |
| --- | --- | --- | --- |
| (001, | 'Windows для чайников', 15000, 'Сормовский', 400), | | |
| (002, | 'Excel 5.0', 23000, 'Сормовский', 360), | | |
| (003, | 'Работа с Visual FoxPro', 32000, 'Нижегородский', 300), | | |
| (004, | 'Программирование в среде Delphi', | 20000, | 'Нижегородский', |
| 100), |  |  |  |
| (005, | 'SQL', 47000, 'Автозаводский', | 89), |  |
| (006, | 'Word 6.0 для Windows', 16000,'Сормовский', | | 200), |
| (007, | 'Твой первый выход в Internet', 15000, 'Советский', | | 140); |

INSERT INTO purchase VALUES

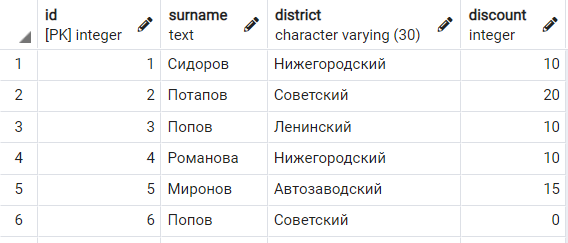
(10011, 'Январь', 001, 006, 003, 2, 64000),

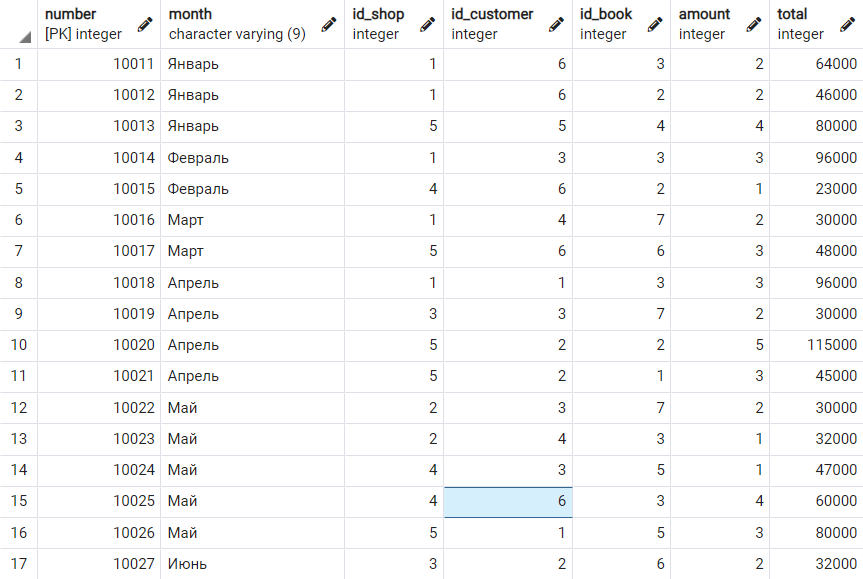
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| (10012, | 'Январь', | | 001, | 006, | 002, | 2, | 46000), |
| (10013, | 'Январь', | | 005, | 005, | 004, | 4, | 80000), |
| (10014, | 'Февраль', | | 001, | 003, | 003, | 3, | 96000), |
| (10015, | 'Февраль', | | 004, | 006, | 002, | 1, | 23000), |
| (10016, | 'Март', 001, 004, | | | 007, | 2, | 30000), | |
| (10017, | 'Март', 005, 006, | | | 006, | 3, | 48000), | |
| (10018, | 'Апрель', | | 001, | 001, | 003, | 3, | 96000), |
| (10019, | 'Апрель', | | 003, | 003, | 007, | 2, | 30000), |
| (10020, | 'Апрель', | | 005, | 002, | 002, | 5, | 115000), |
| (10021, | 'Апрель', | | 005, | 002, | 001, | 3, | 45000), |
| (10022, | 'Май', | 002, 003, | | 007, | 2, | 30000), | |
| (10023, | 'Май', | 002, 004, | | 003, | 1, | 32000), | |
| (10024, | 'Май', | 004,003, | | 005, | 1, | 47000), | |
| (10025, | 'Май', | 004,006, | | 003, | 4, | 60000), | |
| (10026, | 'Май', | 005, 001, | | 005, | 3, | 80000), | |
| (10027, | 'Июнь', | 003, | | 002, | 006, | 2, 32000); | |

# Используя оператор SELECT создать запрос для вывода всех строк каждой таблицы. Проверить правильность ввода. При необходимости произвести коррекцию значений операторами INSERT, UPDATE, DELETE.

SELECT \* FROM book;

SELECT \* FROM customer;



SELECT \* FROM purchase;

SELECT \* FROM shop;

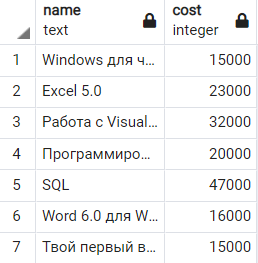


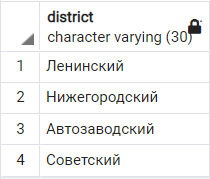
# Создать запросы для вывода:

1. **всех различных названий и стоимостей книг;**

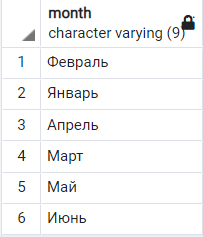
# всех различных районов, в которых проживают покупатели;

1. **всех различных месяцев, когда производились покупки.**

SELECT name, cost FROM book;

SELECT DISTINCT district FROM customer;

SELECT DISTINCT month FROM purchase;

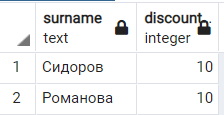


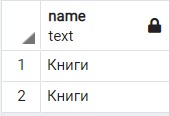
# Создать запросы для получения информации о:

1. **фамилиях и размере скидки всех покупателей, проживающих в Нижегородском районе;**

# названиях магазинов Сормовского или Советского районов;

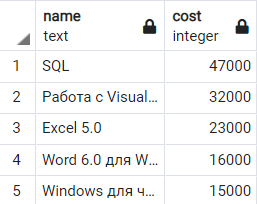
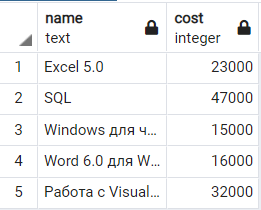
1. **Названиях и стоимости книг, в которых встречается слово Windows, или стоящих более 20000 руб. Вывод результатов организовать по названию и убыванию цены книг.**

SELECT surname, discount FROM customer WHERE district = 'Нижегородский';

SELECT name FROM shop WHERE district = 'Сормовский' or district = 'Советский';

SELECT name, cost FROM book WHERE name LIKE '%Windows%' or cost > 20000 ORDER BY name, cost DESC;

(если нужна сортировка именно по цене, то надо убрать name)



# Для каждой покупки вывести следующие данные:

1. **фамилию покупателя и название магазина, где производилась покупка;**

# дату, фамилию покупателя, скидку, название и количество купленных книг.

SELECT customer.surname, shop.name FROM purchase JOIN customer ON (purchase.id\_customer = customer.id) JOIN shop ON (purchase.id\_shop = shop.id);



SELECT purchase.month, customer.surname, customer.discount, book.name, purchase.amount FROM purchase

JOIN customer ON (purchase.id\_customer = customer.id) JOIN book ON (purchase.id\_book = book.id);



# Определить:

1. **номер заказа, фамилию покупателя и дату для покупок в которых было продано книг на сумму не меньшую чем 60000 руб.**

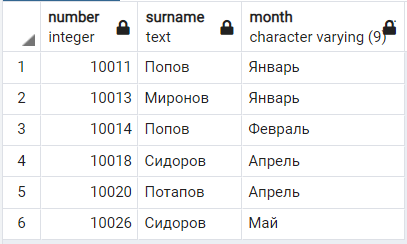
# покупки, сделанные покупателем в своем районе не ранее марта месяца. Вывести фамилию покупателя, район, дату. Произвести сортировку;

1. **магазины, расположенные в любом районе, кроме Автозаводского, где покупали книги те, у кого скидка от 10 до 15 %;**

# данные по покупке книг (название, район складирования, количество), приобретенных в районе складирования и содержащихся в запасе более 10 штук. Включить данные о стоимости и отсортировать по возрастанию.

SELECT purchase.number, customer.surname, purchase.month FROM purchase

JOIN customer ON (purchase.id\_customer = customer.id) WHERE purchase.total > 60000;

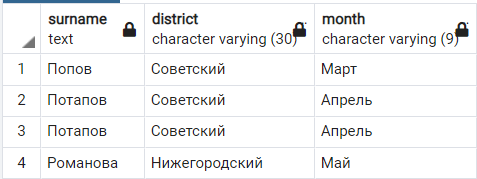


SELECT customer.surname, customer.district, purchase.month FROM purchase

JOIN customer ON (purchase.id\_customer = customer.id) JOIN shop ON (purchase.id\_shop = shop.id)

WHERE customer.district = shop.district AND purchase.month != 'Январь' AND purchase.month != 'Февраль'

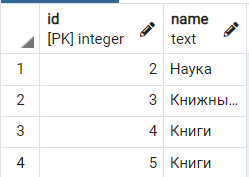
ORDER BY customer.surname;



SELECT DISTINCT shop.id, shop.name FROM shop JOIN purchase ON (shop.id = purchase.id\_shop)

JOIN customer ON (purchase.id\_customer = customer.id)

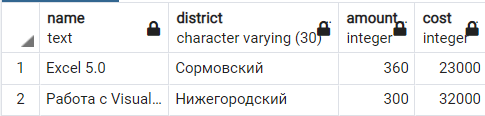
WHERE shop.district != 'Автозаводский' AND customer.discount BETWEEN 10 AND 15;



SELECT book.name, book.district, book.amount, book.cost FROM book JOIN purchase ON (book.id = purchase.id\_book)

JOIN shop ON (purchase.id\_shop = shop.id)

WHERE book.district = shop.district AND book.amount > 10 ORDER BY book.cost;



# Создать запрос для модификации всех значений столбца с суммарной величиной покупки, чтобы он содержал истинную сумму, оплачиваемую покупателем (с учетом скидки). Вывести новые значения.

UPDATE purchase

SET total = purchase.total - purchase.total \* customer.discount \* 0.01 FROM customer

WHERE purchase.id\_customer = customer.id;

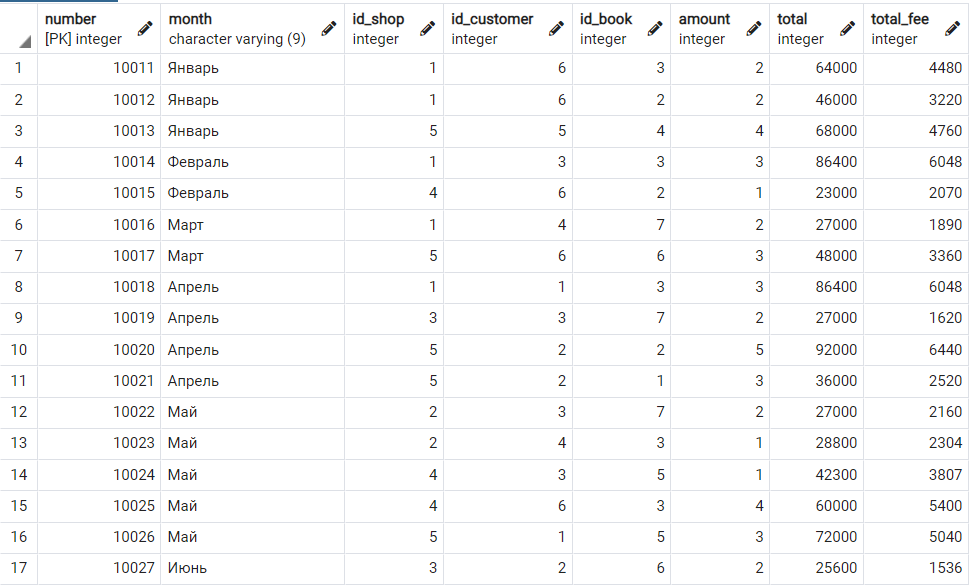
# Расширить таблицу с данными о покупке столбцом, содержащим величину комиссионных, получаемых магазином. Создать запрос для ввода конкретных значений во все строки таблицы покупок.

ALTER TABLE purchase ADD total\_fee INTEGER;

UPDATE purchase

SET total\_fee = purchase.total \* shop.fee \* 0.01 FROM shop

WHERE purchase.id\_shop = shop.id; Таблица purchase после изменений:



***Уровень 2***

# Используя операцию IN (NOT IN) реализовать следующие запросы:

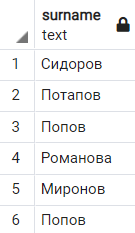
1. **найти покупателей, которые не покупали книг в магазинах Нижегородского района в июне месяце;**

# найти покупателей, покупавших книги в мае на сумму, меньшую чем купил Потапов в том же месяце;

1. **реализовать запросы заданий 7.а, 7.с.**

a)

SELECT customer.surname FROM customer WHERE customer.id NOT IN (SELECT purchase.id\_customer FROM purchase WHERE purchase.month = 'Июнь' AND purchase.id\_shop IN

(SELECT shop.id FROM shop WHERE shop.district = 'Нижегородский'));

b)

SELECT customer.surname FROM customer WHERE customer.id IN (SELECT purchase.id\_customer FROM purchase

WHERE customer.surname != 'Потапов' and purchase.month = 'Май' GROUP BY customer.surname, purchase.id\_customer HAVING sum(purchase.total)

< ANY(SELECT sum(purchase.total) FROM purchase

WHERE purchase.id\_customer IN (SELECT customer.id FROM customer WHERE customer.surname = 'Потапов' and purchase.month = 'Май' GROUP BY customer.surname, customer.id)))



c)

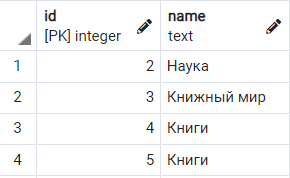
SELECT purchase.number, purchase.month FROM purchase

WHERE purchase.id\_customer IN (SELECT customer.id FROM customer WHERE purchase.total > 60000);



SELECT DISTINCT shop.id, shop.name FROM shop WHERE shop.id IN (SELECT purchase.id\_shop FROM purchase

JOIN customer ON (purchase.id\_customer = customer.id) WHERE shop.district != 'Автозаводский' AND customer.discount BETWEEN 10 AND 15);



# Используя операции ALL-ANY реализовать следующие запросы:

1. **определить покупателя, имеющего минимальную скидку среди тех, кто покупал книги на сумму не менее 50000руб.**

# найти покупателя, покупавшего самое большое количество книг;

1. **запрос задания 7.b;**

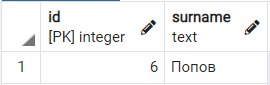
# какой из покупателей не покупавший книг в магазинах своего района, делал покупки на минимальную сумму.

a)

SELECT customer.id, customer.surname FROM customer

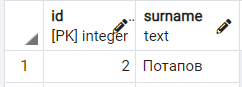
WHERE customer.discount <= ALL(SELECT customer.discount FROM customer JOIN purchase ON (customer.id = purchase.id\_customer)

WHERE purchase.total >= 50000)



b)

SELECT customer.id, customer.surname FROM customer JOIN purchase ON (customer.id = purchase.id\_customer) WHERE purchase.amount >= ALL(SELECT purchase.amount FROM purchase)

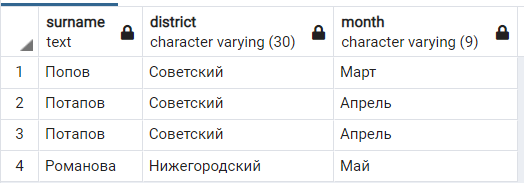


c)

SELECT customer.surname, customer.district, purchase.month FROM purchase

JOIN customer ON (purchase.id\_customer = customer.id) WHERE purchase.id\_shop

= ANY(SELECT shop.id FROM shop WHERE customer.district = shop.district AND purchase.month != 'Январь' AND purchase.month != 'Февраль') ORDER BY customer.surname;



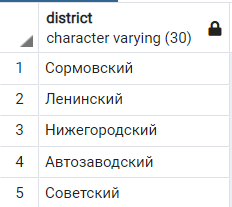
d)

SELECT customer.id, customer.surname, customer.district, shop.district, purchase.total FROM customer

JOIN purchase ON (customer.id = purchase.id\_customer) JOIN shop ON (purchase.id\_shop = shop.id)

WHERE customer.district != shop.district AND purchase.total <= ALL(SELECT purchase.total FROM purchase);

# Используя операцию UNION получить районы проживания покупателей и районы складирования книг.

SELECT customer.district FROM customer UNION SELECT book.district FROM book;

# Используя операцию EXISTS ( NOT EXISTS ) реализовать ниже следующие запросы. В случае, если для текущего состояния БД

**запрос будет выдавать пустое множество строк, требуется указать какие добавления в БД необходимо провести.**

# какой покупатель покупал все книги в магазине “Наука” или “Знание”;

1. **найти покупателей, покупавших книги во всех магазинах своего района до декабря;**

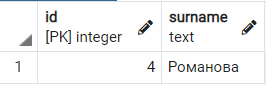
# определить покупателей, покупавших все книги, не продающиеся в магазине с максимальным значением комиссионных;

1. **найти среди покупателей тех, кто не покупал в мае книг со ценой более 25000 руб. в магазинах с максимальным размером комиссионных.**

SELECT customer.id, customer.surname FROM customer WHERE EXISTS (SELECT \* FROM purchase JOIN shop ON (purchase.id\_shop = shop.id) WHERE shop.name = 'Наука' AND customer.id = purchase.id\_customer AND EXISTS

(SELECT \* FROM purchase JOIN shop ON (purchase.id\_shop = shop.id) WHERE shop.name = 'Знание' AND customer.id = purchase.id\_customer

AND NOT EXISTS (SELECT \* FROM purchase JOIN shop ON (purchase.id\_shop = shop.id)

WHERE shop.name != 'Наука' AND shop.name != 'Знание' AND customer.id = purchase.id\_customer)))

#c

SELECT DISTINCT purchase.id\_book, book.name FROM purchase JOIN

book ON (purchase.id\_book = book.id)

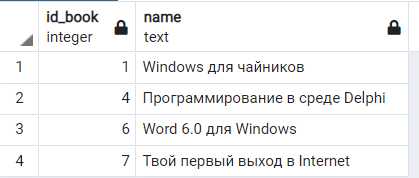
WHERE EXISTS (SELECT \* FROM book book1 WHERE NOT EXISTS

(SELECT \* FROM book book2 JOIN purchase ON (book2.id = purchase.id\_book)

WHERE EXISTS (SELECT \* FROM shop shop1 WHERE NOT EXISTS

(SELECT \* FROM shop shop2 WHERE shop1.fee < shop2.fee)

AND purchase.id\_shop = shop1.id) AND book1.id = book2.id) AND book1.id = purchase.id\_book)

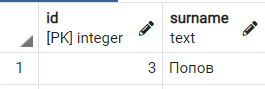


#d

SELECT customer.id, customer.surname FROM customer JOIN purchase ON (customer.id = purchase.id\_customer) JOIN book ON (purchase.id\_book

= book.id)

WHERE purchase.month = 'Май' AND book.cost < 25000 AND EXISTS (SELECT shop1.name FROM shop shop1 WHERE NOT EXISTS (SELECT \* FROM shop shop2 WHERE shop1.fee < shop2.fee AND customer.id = purchase.id\_customer))



# Реализовать запросы с использованием агрегатных функций:

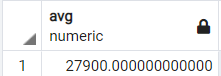
1. **получить среднюю стоимость покупок, сделанных в магазинах Нижегородского района;**

# найти количество покупателей, покупавших книги в магазине “Наука”;

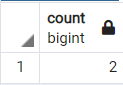
1. **найти покупателей имеющих скидку ниже средней;**

# определить магазины, в которых покупало книги больше покупателей чем в магазине “Наука”.

SELECT avg(purchase.total) FROM purchase JOIN shop ON (purchase.id\_shop = shop.id) WHERE shop.district = 'Нижегородский'

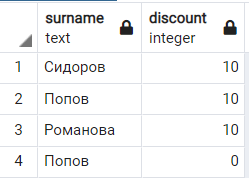


SELECT count(distinct purchase.id\_customer) FROM purchase JOIN shop ON (purchase.id\_shop = shop.id) WHERE shop.name = 'Наука';



SELECT customer.surname, customer.discount FROM customer GROUP BY customer.surname, customer.discount

HAVING customer.discount < ANY(SELECT avg(customer.discount) FROM customer)



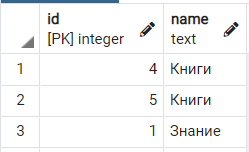
SELECT shop.id, shop.name FROM shop JOIN purchase ON (shop.id = purchase.id\_shop) GROUP BY shop.id, shop.name

HAVING count(purchase.id\_customer)

> ANY (SELECT count(purchase.id\_customer) FROM purchase

JOIN shop ON (purchase.id\_shop = shop.id) WHERE shop.name =

'Наука')



# Используя средства группировки реализовать следующие запросы:

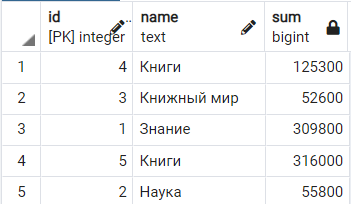
1. **вывести данные по суммарной стоимости книг, купленных в каждом магазине;**

# вывести отчет о суммарной стоимости всех купленных книг по районам, где расположены магазины;

1. **получить сводную информацию о сумме всех покупок, произведенных каждым покупателем;**

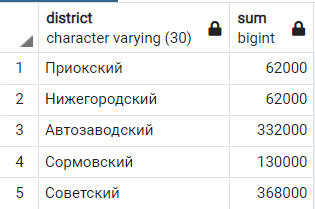
# определить для каждого дня недели количество книг, купленных покупателями не из Советского района .

SELECT shop.id, shop.name, sum(purchase.total) FROM purchase JOIN shop ON (purchase.id\_shop = shop.id)

GROUP BY shop.id, shop.name

# #b

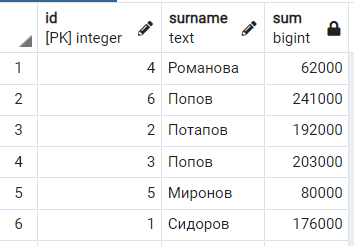
SELECT shop.district, sum(purchase.total) FROM purchase JOIN shop ON (purchase.id\_shop = shop.id) GROUP BY shop.district



SELECT shop.district, sum(purchase.total) FROM purchase RIGHT OUTER JOIN shop ON (purchase.id\_shop = shop.id) GROUP BY shop.district



# #c

SELECT customer.id, customer.surname, sum(purchase.total) FROM purchase JOIN customer ON (purchase.id\_customer = customer.id) GROUP BY customer.id, customer.surname

SELECT customer.id, customer.surname, sum(purchase.total) FROM purchase RIGHT OUTER JOIN customer ON (purchase.id\_customer = customer.id) GROUP BY customer.id, customer.surname

# #d

SELECT purchase.month, sum(purchase.amount) FROM purchase JOIN customer ON (purchase.id\_customer = customer.id)

WHERE customer.district != 'Советский' GROUP BY purchase.month

